**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA ĐIỆN TỬ - TIN HỌC**



A close up of a sign

Description automatically generated

**BÁO CÁO CHUYÊN ĐỀ**

**GIT / GITHUB**

**Sinh viên** : Nguyễn Anh Thi – 0306171199

Nguyễn Hiếu Luân – 0306171163

Lê Đình Bảo Duy – 0306171121

**Môn** :Công cụ và môi trường phát triển phần mềm

**Giáo viên** : Tôn Long Phước

**Lớp** : CĐTH17PMB

Mục lục

[I. Giới thiệu về GIT 4](#_Toc20429780)

[1. Lịch sử 4](#_Toc20429781)

[2. Git là gì ? 4](#_Toc20429782)

[3. Các khái niệm cơ bản trên Git 5](#_Toc20429783)

[a) Version Control System - VCS 5](#_Toc20429784)

[b) Repository 5](#_Toc20429785)

[c) Commit 7](#_Toc20429786)

[d) Branch 7](#_Toc20429787)

[e) Marge 8](#_Toc20429788)

[f) Push 8](#_Toc20429789)

[g) Pull 9](#_Toc20429790)

[h) Check out 9](#_Toc20429791)

[i) Conflict 9](#_Toc20429792)

[j) Clone 9](#_Toc20429793)

[k) Working tree và Index 10](#_Toc20429794)

[l) Git Remote 11](#_Toc20429795)

[4. Tại sao nên sử dụng Git? 11](#_Toc20429796)

[II. Giới thiệu về GitHub 12](#_Toc20429797)

[1. Lịch sử 12](#_Toc20429798)

[2. GitHub là gì? 12](#_Toc20429799)

[3. Phân biệt Git và GitHub 12](#_Toc20429800)

[4. Các khái niệm cơ bản trên GitHub 13](#_Toc20429801)

[a) Fork 13](#_Toc20429802)

[b) Pull Request 13](#_Toc20429803)

[5. Những tính năng hữu ích của GitHub 13](#_Toc20429804)

[a) Kéo và thả code với Gist 13](#_Toc20429805)

[b) Tạo một thư mục thông qua giao diện web 14](#_Toc20429806)

[c) Sử dụng Git URL Shortener 14](#_Toc20429807)

[d) Tìm kiếm tập tin và file 15](#_Toc20429808)

[e) Sử dụng GitHub Emoji 15](#_Toc20429809)

[f) Map, CSV và 3D Rending 15](#_Toc20429810)

[6. Cách mà GitHub kiếm tiền 15](#_Toc20429811)

[7. Cách quản lý phiên bản của Github 16](#_Toc20429812)

[8. Sơ đồ hoạt động của GitHub 17](#_Toc20429813)

[9. GitHub mang lại cho lập trình viên những gì? 17](#_Toc20429814)

[III. Tổng kết 18](#_Toc20429818)

# Mục lục hình

[Hình 1 5](#_Toc20429931)

[Hình 2 6](#_Toc20429932)

[Hình 3 6](#_Toc20429933)

[Hình 4 7](#_Toc20429934)

[Hình 5 8](#_Toc20429935)

[Hình 6 8](#_Toc20429936)

[Hình 7 9](#_Toc20429937)

[Hình 8 10](#_Toc20429938)

[Hình 9 14](#_Toc20429939)

[Hình 10 15](#_Toc20429940)

[Hình 11 16](#_Toc20429941)

[Hình 12 17](#_Toc20429942)

# Giới thiệu về GIT

# Lịch sử

Nhân Linux là một dự án mã nguồn mở trong một phạm vi lớn. Trong phần lớn thời gian phát triển của Linux (1991 – 2002), các thay đổi của phần mềm được truyển đi dưới dạng các bản vá và các tập tin lưu trữ. Vào năm 2002, dự án nhân Linux bắt đầu sử dụng DVCS (Distributed Version Control System – Hệ thống quản lý phiên bản phân tán) tên là BitKeeper.

Vào năm 2005, sự hợp tác giữa cộng đồng phát triển nhân Linux và công ty BitKeeper bị phá vỡ, và công cụ đó không còn được cung cấp miễn phí nữa. Điều này đã thúc cộng đồng phát triển nhân Linux (hay chính xác hơn là Linus Torvalds – người sáng lập Linux, phát triển công cụ của riêng họ dựa trên những bài học từ việc sử dụng BitKeeper. Và Git ra đời, một công cụ nhanh, thiết kế đơn giản, phân tán toàn diện, có khả năng xử lí các dự án lớn giống như nhân Linux

# Git là gì ?

Git là một hệ thống quả lý phiên bản phân tán (Distributed Version Control System – DVCS). Trên Git, ta có thể lưu trạng thái của file dưới dạng lịch sử cập nhật. Vì thế, có thể đưa file đã chỉnh sửa một lần về trạng thái cũ hay có thể biết được file đã được chỉnh sửa chỗ nào.

**GitHub** là sự kết hợp giữa 2 từ, Git – hệ thống quản lý dự án và phiên bản code và Hub – một mạng xã hội cho lập trình viên. Khi sử dụng GitHub, ngoài các công việc chính như tạo Branch, tạo Pull Request người dùng có thể theo dõi, tương tác với người khác như một mạng xã hội thông thường.

Git sử dụng mô hình phân tán, mỗi nơi lưu trữ source ta gọi là 1 repo (repository), các lập trình viên sẽ tạo một repo tại máy của mình. Điều đó có nghĩa là nếu có 3 người A,B,C cùng làm việc trong 1 project. Thì bản thân repo trên máy của người A, người B, và người C có thể kết nối được với nhau.

Hình 1

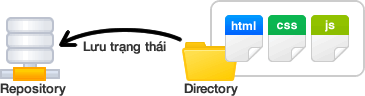
# Các khái niệm cơ bản trên Git

# Version Control System - VCS

VCS là một hệ thống lưu giữ các phiên bản của mã nguồn của sản phẩm phần mềm, giúp các lập trình viên có thể dễ dàng lấy lại phiên bản mong muốn. Có nhiều version control system khác như CSV, SVN, TeamVS,... Git chính là một trong những hệ thống quản lý phiên bản đó.

# Repository

Repository hay được gọi tắt là Repo, đơn giản là nơi chứa tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của toàn bộ project.

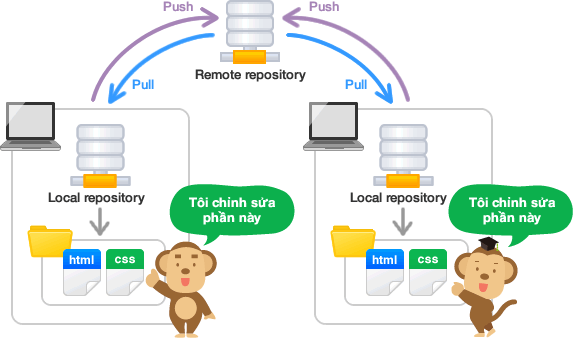


Hình 2

Repository của Git được phân thành 2 loại là remote repository và local repository.

* Remote repository: Là repo để chia sẻ giữa nhiều người và được lưu trên server.
* Local repository: Là repo trên máy người dùng

Do repository phân thành 2 loại là local và remote nên với những công việc bình thường thì có thể sử dụng local repository. Khi muốn public nội dung công việc mà mình đã làm trên local repository, thì ta sẽ upload lên remote repository rồi public. Thêm nữa, thông qua remote repository người dùng cũng có thể lấy về nội dung thay đổi của người khác.

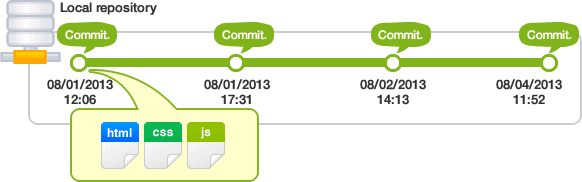


Hình 3

# Commit

Để ghi lại việc thêm/thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit.

Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước với trạng thái hiện tại. Bằng việc lần theo commit này từ trạng thái mới nhất thì có thể biết được lịch sử thay đổi trong quá khứ hoặc nội dung thay đổi đó.



Hình 4

# Branch

Branch là các nhánh phát triển của repo, được dùng để phát triển tính năng mới mà không làm ảnh hưởng đến code hiện tại.

Mỗi branch giống như một không gian làm việc khác nhau, branch có thể được chia tách cũng như sát nhập dễ dàng.

Mỗi repo sẽ có một branch chính là master, là nhánh “mặc định” khi tạo một repository. Nhánh master thông thường là nhánh chính của ứng dụng.

Ví dụ: bạn thử nghiệm một tính năng mới và muốn không ảnh hưởng đến code chính bạn có thể tạo một nhánh mới và sau khi xong sẽ hợp nhất lại với nhánh master. Việc hợp nhất 2 nhánh lại được gọi là merge.



Hình 5

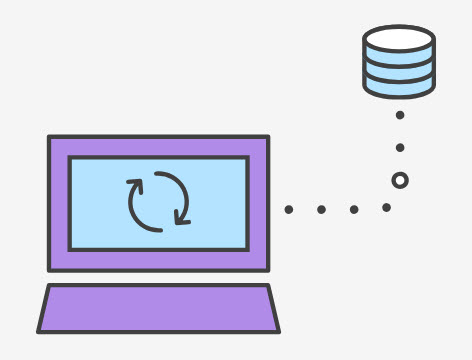
# Marge

Marge là hành động hợp nhất một nhánh phát triển vào nhánh khác

Ví dụ: Bạn phát triển xong 1 tính năng, đã test/ kiểm và thấy nó hoàn chỉnh, có thể tích hợp vào phần mềm thì bạn sẽ tiến hành merge code. Sau khi merge có thể giữ lại 1 trong 2 branch hoặc cả 2. Yêu cầu trước khi merge: phải push hết các commit lên branch.

# Push

Push là hành động upload một commit hoặc branch lên remote repo. Sau khi upload lên thì các thành viên của team có thể thấy và đồng bộ code xuống máy.

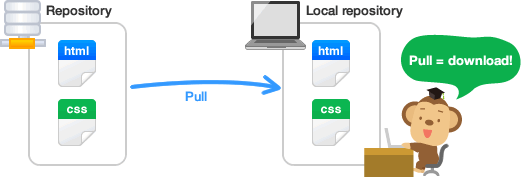


Hình 6

# Pull

Pull  là hành động download các thay đổi xuống local repo.

Ví dụ: trong khi bạn đang code trên một file thì một người bạn trong nhóm của bạn cũng code trên một file khác cùng branch, người bạn đó hoàn thành công việc, commit và push lên remote repo. Lúc này bạn muốn lấy những thay đổi mà người bạn của bạn đã thực hiện thì bạn sẽ thực hiện hành động Pull xuống.



Hình 7

# Check out

Check out là hành động chuyển sang làm việc trên một branch khác. Trước khi chuyển branch thì người dùng phải lưu lại trạng thái của branch hiện tại bằng cách commit.

# Conflict

Conflict là trường hợp có nhiều sự thay đổi trong cùng 1 dòng code khi merge và máy không thể tự quyết định cái nào là đúng. Lúc này người dùng phải tự quyết định giữ lại dòng code nào. Các xung đột này phải được giải quyết bằng cách nhìn vào file bị conflict và tự quyết định dòng code nào giữ lại, dòng nào xóa bỏ.

# Clone

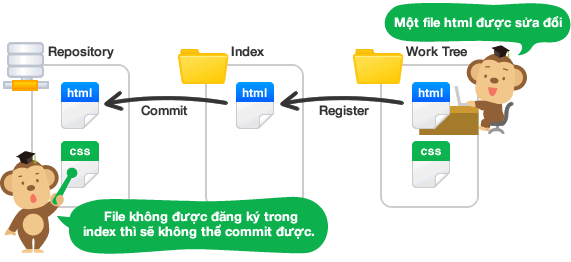
 Clone là hành động tạo bản sao của remote repo từ máy chủ về máy mình để có thể lập trình và phát triển.

Đây là điểm khác biệt của Git so với một số hệ thống quản lý phiên bản mã nguồn khác vì clone là tạo ra một bản sao của gần như tất cả những gì của repository mà máy chủ đang lưu trữ. Người dùng sẽ có được tất cả lịch sử đã xảy ra trên repository và hoàn toàn có thể quay lại, undo lại từ bất kỳ thời điểm commit nào. Và một điểm nữa là nếu ổ cứng máy chủ bị hư, người dùng có thể sử dụng bất kỳ bản sao trên bất kỳ máy khách nào để khôi phục lại trạng thái của máy chủ.

# Working tree và Index

Trên Git, những thư mục được đặt trong sự quản lý của Git mà mọi người đang thực hiện công việc trong thực tế được gọi là working tree.

Và trên Git, giữa repository và working tree tồn tại một nơi gọi là index. Index là nơi để chuẩn bị cho việc commit lên repository.



Hình 8

Trên Git, khi đã thực hiện commit thì trạng thái sẽ không được ghi trực tiếp trong repository từ working tree, mà sẽ ghi trạng thái đã được thiết lập của index được xây dựng ở giữa đó. Vì thế, để ghi lại trạng thái của file bằng commit thì trước hết cần thông báo file trong index.

# Git Remote

Để kết nối được với một repo khác người ta sử dụng một khái niệm gọi là remote.

Trên thực tế khi làm việc với nhau thì không như vậy, vì không phải máy ai cũng cài một “git server” để người khác kết nối được với mình. Thông thường thì chúng ta sẽ sử dụng một repo chung và các máy kết nối vào repo đó.

Có 2 “git repo server” được sử dụng nhiều là github.com và bitbucket.org.

Trên thực tế khi có 2 người cùng làm việc với 1 project thì thông thường sẽ tạo một repo trên github hoặc bitbucket và repo trên máy người A sẽ kết nối với repo trên github và máy người B cũng kết nối với repo trên github/bitbucket. Từ đó source code của người A và người B sẽ được đồng bộ với nhau thông qua repo trên github/bitbucket.

# Tại sao nên sử dụng Git?

Có rất nhiều lợi thế để lập trình viên nên sử dụng Git trong việc lập trình ngay từ hôm nay, bất kể là lập trình cái gì đi chăng nữa.

* Git dễ sử dụng, an toàn và nhanh chóng.
* Quản lý source code dễ dàng chuyên nghiệp
* Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).
* Code không giới hạn khoảng cách giữa các thành viên trong team, lập trình viên có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa.
* Dễ dàng trong việc triển khai sản phẩm.
* Khi gặp lỗi có thể dễ dàng Backup lại phiên bản trước
* Tự tin hơn khi thử nghiệm những ý tưởng mới
* Và nhiều hơn thế nữa.

# Giới thiệu về GitHub

# Lịch sử

Sự phát triển của GitHub bắt đầu vào ngày 19 tháng 10 năm 2007. Trang web được đưa ra vào tháng 4 năm 2008 do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett thực hiện sau khi nó đã được hoàn thành một vài tháng trước đó.

Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Vào ngày 4 tháng 6 năm 2018, Microsoft đã thông báo việc đạt được thỏa thuận mua lại GitHub với giá 7.5 tỷ Đô la Mỹ.

# GitHub là gì?

GitHub là một dịch vụ cung cấp [kho lưu trữ mã nguồn](https://vi.wikipedia.org/w/index.php?title=Kho_l%C6%B0u_tr%E1%BB%AF_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)) dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản.

Trái tim của GitHub chính là Git. GitHub là một dịch vụ lưu trữ sử dụng Git, nhưng bổ sung thêm nhiều tính năng riêng của nó.

# Phân biệt Git và GitHub

Chúng ta đã nghe tới Git và có thể đã dùng GitHub nhưng chúng ta đa phần vẫn nhầm lẫn Git vs GitHub là một bởi vì cứ nói đến Git là nghĩ tới GitHub. Đó là một sự hiểu lầm vì Git như đã giải thích ở trên đó là tên gọi của một mô hình hệ thống, các máy tính có thể clone lại mã nguồn từ một repository , còn GitHub là tên của một công ty cũng cấp dịch vụ máy chủ repository công cộng, mỗi người có thể truy cập vào website trang chủ để tạo tài khoản trên đó và tạo ra kho chứa source của riêng mình khi làm việc, nó cũng cung cấp công cụ kiểm soát truy cập và một số tính làm việc nhóm, chẳng hạn như wiki và các công cụ quản lý task cơ bản.

# Các khái niệm cơ bản trên GitHub

Ngoài các khái niệm của Git, GitHub còn có thêm 1 số khái niệm riêng như là fork và pull request.

# Fork

Chức năng hàng đầu của GitHub là “forking” – sao chép một kho lưu trữ từ tài khoản của người dùng này sang tài khoản khác.

Điều này cho phép bạn tham gia vào một dự án mà bạn không có quyền ghi và sửa đổi trực tiếp lên đó bằng tài khoản của bạn. Nếu bạn thực hiện các thay đổi và muốn chia sẻ nó, bạn có thể gửi một thông báo được gọi là “pull request” tới chủ sở hữu ban đầu của dự án. Sau đó, chỉ cần một cú click chuột của chủ sở hữu vào một button, thay đổi bạn tạo ra sẽ được “merge” vào repo gốc.

# Pull Request

Pull Request (viết tắt là PR) sẽ để cho bạn nói với người khác về các thay đổi bạn đã đẩy lên kho Github. Một khi pull request được gửi, người nào quan tâm có thể xem xét lại các thay đổi, hoặc thảo luận các sửa đổi tiềm năng, và có thể theo đó đẩy tiếp các commit của họ nếu cần thiết.

Khi bạn gửi một “pull request”, người quản lý dự án có thể xem hồ sơ của bạn, bao gồm tất cả các đóng góp của bạn trên GitHub. Nếu bản vá của bạn được chấp nhận, bạn sẽ nhận được điểm và nó sẽ hiển thị trong tiểu sử của bạn.

# Những tính năng hữu ích của GitHub

# Kéo và thả code với Gist

Gist là 1 phần riêng biệt của Github, cho phép lưu trữ các đoạn code. Cũng có thể duyệt và tìm một số lượng lớn các đoạn mã của nhiều ngôn ngữ khác nhau. Sử dụng Gist hoàn toàn dễ dàng và phải trực quan. Chỉ cần kéo và thả các tệp từ máy tính vào Gist, các đoạn code trong các tệp sẽ được sao chép ngay lập tức. Nó nhanh chóng và tiết kiệm rất nhiều thời gian.

# Tạo một thư mục thông qua giao diện web

Trong khi nhiều người có thể quản lý repo Github thông qua ứng dụng Github miễn phí. Ngoài ra, Github cũng đã xây dựng những gì họ gọi là WebFlow. Nó cho phép chúng ta quản lý repo thông qua giao diện web của Github.

# Sử dụng Git URL Shortener

Hiện nay, mọi hoạt động thường ngày của mọi người đều được cập nhật lên mạng xã hội từ hình ảnh, trạng thái… Và người dùng GitHub cũng không ngoại lệ. Nếu bạn là người dùng Github, bạn cũng có thể muốn chia sẻ repo Github của mình. Tuy nhiên, URL của repo đôi khi quá dài để chia sẻ, ví dụ như Twitter chỉ chấp nhận URL chứa tối đa 140 ký tự. Git.io sẽ rút ngắn URL của repo Github của bạn.



Hình 9

# Tìm kiếm tập tin và file

Bên cạnh việc tạo các tệp mới, người dùng cũng có thể điều hướng và tìm kiếm nhanh chóng qua các tệp trong bất kỳ repo nào. Tính năng này rất hữu ích nhưng lại không rõ ràng vì nó bị ẩn và chỉ hoạt động khi dùng phím tắt.

Nhấn phím T để kích hoạt Trình tìm kiếm tệp. Nhấn mũi tên lên và mũi tên xuống để nhảy qua các tệp lên và xuống. Hoặc, nhập tên tệp để chọn một tệp cụ thể

# Sử dụng GitHub Emoji

Cũng giống như các mạng xã hội khác GitHub cũng có comment các đoạn code trên GitHub. Và đặc biệt Github cũng tích hợp 1 số Emoji vô cùng dễ thương.



Hình 10

# Map, CSV và 3D Rending

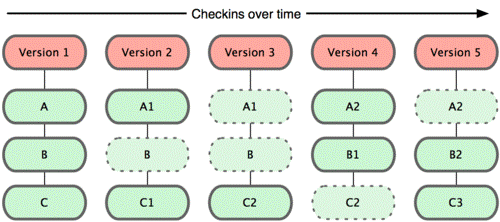
Gihub hỗ trợ file **CSV**. Nếu bạn include một tệp .csv, Github sẽ hiển thị tệp CSV thành định dạng dữ liệu bảng. Nó thậm chí còn cho phép tìm kiếm thông qua nó. Ngoài CSV, Github cũng sẽ tự động hiển thị Bản đồ với định dạng **geoJSON** và 3D với phần mở rộng **STL** .

# Cách mà GitHub kiếm tiền

Bên cạnh các kho lưu trữ công khai (public), GitHub cũng bán các kho lưu trữ riêng tư (private) và tại chỗ ([on-premises)](https://www.webopedia.com/TERM/O/on-premises.html) cho các doanh nghiệp. (On-premises là kiểu mà kho lưu trữ sẽ được đặt tại chính công ty khách hàng của GitHub, GitHub cung cấp phần cứng và hỗ trợ kỹ thuật).

# Cách quản lý phiên bản của Github

GitHub sử dụng kỹ thuật quản lý phiên bản của Git. Do đó, nó coi dữ liệu của nó giống như một tập hợp các ảnh (snapshot) của một hệ thống tập tin nhỏ. Mỗi lần commit, GitHub sẽ chụp một bức ảnh (snapshot) ghi lại nội dung của tất cả các tập tin tại thời điểm đó và tạo ra một tham chiếu tới ảnh đó. Để hiệu quả hơn, nếu như tập tin không có sự thay đổi nào, GitHub không lưu trữ tập tin đó lại một lần nữa mà chỉ tạo một liên kết tới tập tin gốc đã tồn tại trước đó.



Hình 11

Cách thức quản lý này cũng giống như một hệ thống quản lý tập tin thu nhỏ. Do đó, GitHub có thể cung cấp nhiều tính năng, công cụ vô cùng mạnh mẽ.

# Sơ đồ hoạt động của GitHub



Hình 12

# GitHub mang lại cho lập trình viên những gì?

Ngoài một profile [Stackoverflow](http://stackoverflow.com/) ấn tượng thì một tài khoản GitHub với nhiều repositories và tham gia đóng góp vào những project đa dạng khác nhau đem đến nhiều lợi ích đáng kể, thậm chí có thể thay đổi cả sự nghiệp của lập trình viên.

# GitHub giúp chứng tỏ bạn là ai

Chẳng thể phủ nhận những lời hay ý đẹp được viết trong CV là cần thiết. Nhưng dù gì thì đó cũng chỉ là những lời vô thưởng vô phạt, chẳng thể nói lên được gì nhiều. Chỉ có những dòng code nằm trên Github mới là bất diệt. Source code luôn là minh chứng đanh thép để thể hiện bạn là lập trình viên thật thụ hay nghiệp dư.

# GitHub giúp cải thiện khả năng code

Có hàng ngàn hàng vạn cách để học, học trên Github sẽ là một ý kiến không tồi trong thời đại này. Với hàng vạn open source projects, hàng trăm ngàn người đóng góp, hàng tỉ commit mỗi ngày thì chỉ bằng việc xem, so sánh, học tập từ những thay đổi đó đã đem lại cho lập trình viên hàng tá điều hay để cải thiện kỹ năng code của bản thân mình.

# GitHub là một kho tài nguyên tuyệt vời

Với mục GUIDES và HELP, lập trình viên có thể đọc bài để gia tăng thêm nhiều kỹ năng cho mình. Hầu hết những thông tin này đều xoay quanh chủ đề về Git, tất tần tật mọi thứ mà ai cũng có thể nghĩ ra nhưng lại không tìm được thông tin.

Đọc EXPLORE và OPEN SOURCE ở Git để có thể lên trình độ code nhanh nhất. Hoặc lập trình viên cũng có thể tự tạo công việc để mình tự học hỏi. Bằng cách đưa ra một dự án ảo, rồi mời mọi người vào cùng làm. Chỉnh sửa và hoàn thiện là công việc chính.

Từ thông tin cộng đồng, có thể lập trình viên sẽ được hỗ trợ tìm ra lỗi của project, sau đó điều chỉnh hoàn thiện.

Github mang lại cho người dùng những trải nghiệm hoàn toàn hữu ích. Tất cả mọi thông tin đều được lưu trữ trên web, khi cần có thể lấy xuống.

# Tổng kết